Java Refactoring with Eclipse Part–II

While concluding the two-part series on refactoring, let's discuss refactoring operations that can be applied to a sample program, making it cleaner and easier to maintain.

In Part 1 of this article, we applied few refactoring techniques to the sample program. In the second part, we will explore other refactoring operations that you can apply to the same program to make it cleaner and easier to maintain.

Here is a quick recap of what refactoring is. It is the process of changing the internal structure of a program (like removing redundancies, eliminating unused functions and rejuvenating obsolete designs) without changing its external behaviour. The idea is to keep the code cleaner and concise so that it is easier to understand, modify and extend.

The first refactoring technique that we will apply to our sample program is the 'Extract Interface'. We will use this technique to extract an interface for the class *JEmailClient*. To extract an interface from a class, select the class name and click on *Extract Interface* from the *Refactor* menu as shown in Figure 1.

Enter the interface name and select the members to declare in the interface from the Extract Interface pop-up dialog box as shown in Figure 2.

Figures 3 and 4 display the preview and interface implementation dialog boxes, respectively. Figures 3 and 4 show that Eclipse has intelligently extracted the interface for the class *JEmailClient* and updated the references accordingly.

Next, we will rename the field *server* to *smtpServer* using the 'Rename Field' refactoring technique. To rename a field, select the field name and click on *Rename* from the *Refactor* menu. Enter the new name for the field in the pop-up dialog box as shown in Figure 5.

Figure 6 shows the preview dialog. It clearly indicates that Eclipse has renamed the field *server* to *smtpServer* and updated all the references accordingly.

Similarly, change the *send* and *read* method names to *sendSMTP* and *readSMTP,* respectively. To rename a method, select the method name and click on *Rename* from the *Refactor* menu. Enter the new name for the method in the pop-up dialog box as shown in Figure 7.

Figure 8 shows the preview dialog. This shows that Eclipse has changed the method name from *send* to *sendSMTP* and updated all the references automatically.

Now, change the method *read* to *readSMTP* as shown in Figures 9 and 10.

The completely restructured program has been incorporated in the LFY CD #1. The program is now cleaner and easier to understand than before. You should note that we were able to do all this without changing the functionality (external behaviour) of the program. You can now appreciate the power of refactoring with Eclipse!

References

- http://www.refactoring.com/
- www.cs.umanitoba.ca/~eclipse/13-Refactoring.pdf
- http://c2.com/cgi/wiki?WhatIsRefactoring

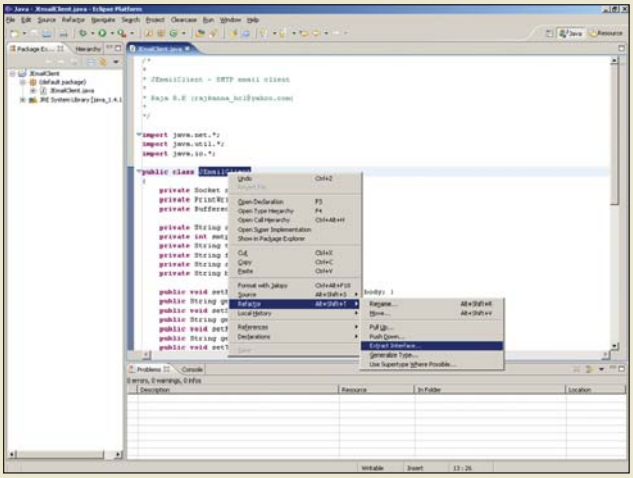

Figure 1: Extract Interface refactoring operation

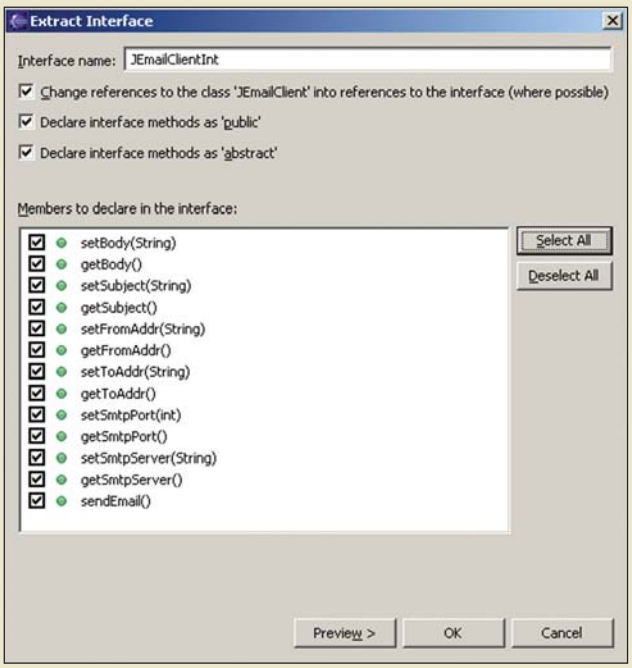

Figure 2: Extract Interface pop-up dialog

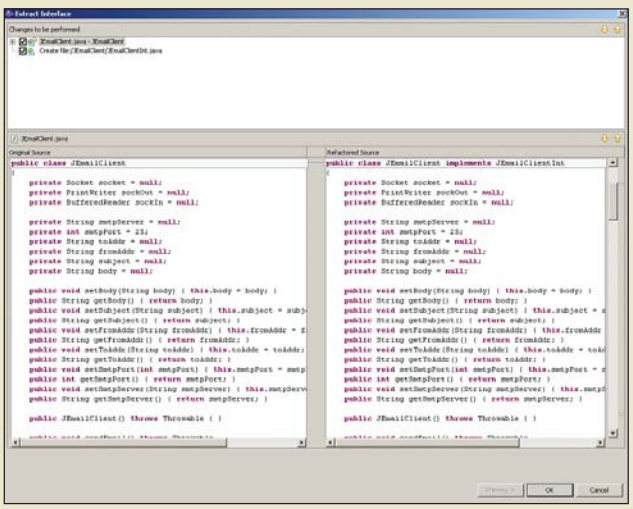

Figure 3: Extract Interface preview dialog

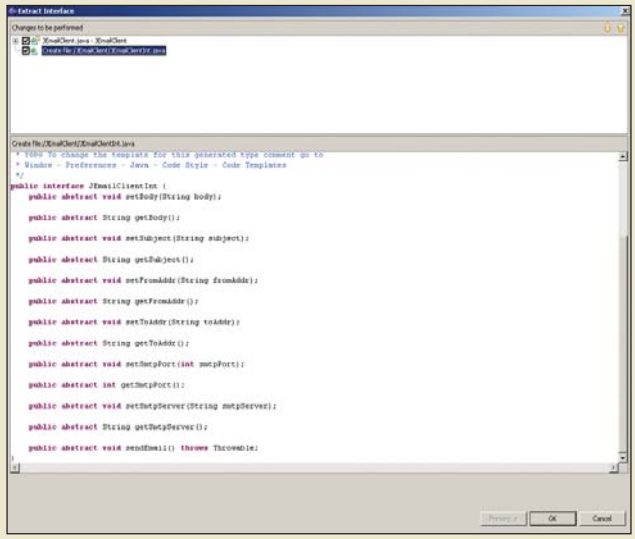

Figure 4: Interface implementation

Figure 5: Rename field pop-up dialog

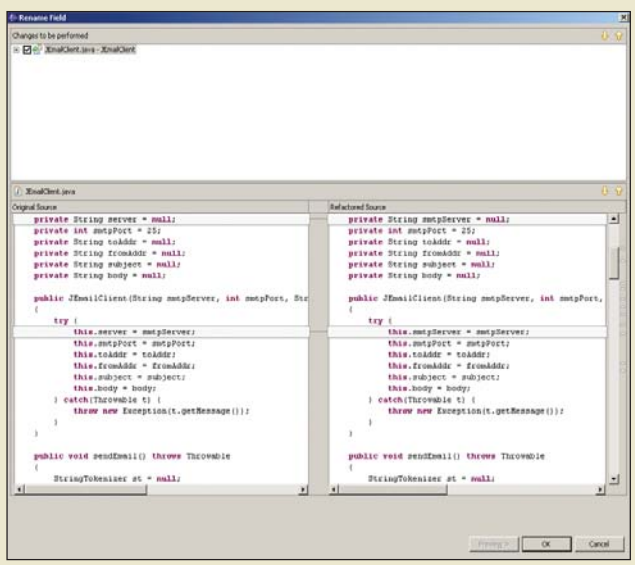

Figure 6: Rename field preview dialog

Figure 7: Rename method pop-up dialog

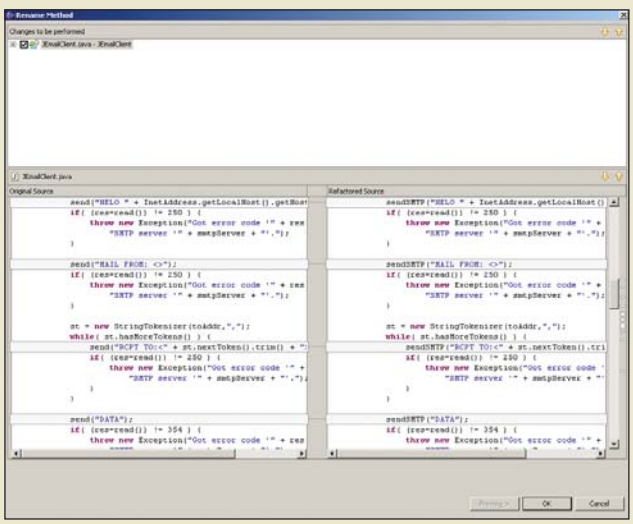

Figure 8: Rename method preview dialog

Figure 9: Rename method pop-up dialog

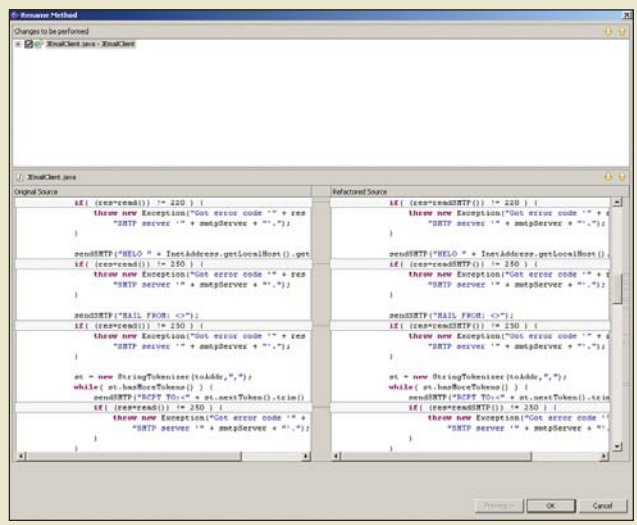

Figure 10: Rename method preview dialog

LFY

By: Raja R K. The author is a lead engineer working with HCL Technologies (Cisco Systems Offshore Development Centre) in Chennai. He can be contacted at: rajark_hcl@yahoo.co.in